PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/50

A1

(11) International Publication Number: WO 97/33243

(43) International Publication Date: 12 September 1997 (12.09.97)

(21) International Application Number: PCT/US97/03473

(22) International Filing Date: 4 March 1997 (04.03.97)

(30) Priority Data:
08/610,924 5 March 1996 (05.03.96) US

(71) Applicant: ELECTRONIC DATA SYSTEMS CORPORATION [US/US]; 5400 Legacy Drive, M/S H3-3A-05, Plano, TX 75024 (US).

(72) Inventors: BARTON, W., Scott; 1090 York Trace, Marietta, GA 30064 (US). ANDREWS, Craig; 1434 Malcolm Drive, Dresher, PA 19025 (US).

(74) Agent: GRIEBENOW, L., Joy; Electronic Data Systems Corporation, 5400 Legacy Drive, M/S H3-3A-05, Plano, TX 75024 (US).

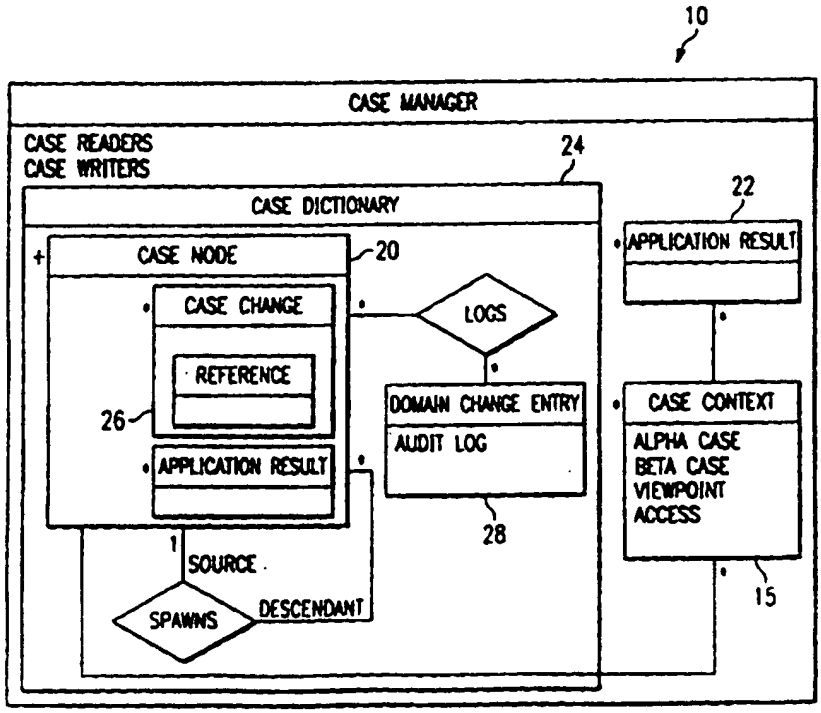
(81) Designated States: AU, CN, CZ, JP, NZ, PL, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A CASE MANAGEMENT SYSTEM AND METHOD FOR MANAGING CASE MODELS OF PHYSICAL SYSTEMS

(57) Abstract

A software case management system and method for managing cases (20) representing models of an operation system where the cases (20) are created by a user (11) having access to a base set of data containing variables. Case manager object (10) receives input from a user to create cases (20) containing data from the base data. Case manager object (10) also creates cases directly from base data set (13) without copying the base data for each case created. Case manager object (10) also receives input from users (11) to create cases (20) containing a combination of unchanged variables from base data set (13) and changed variables from base data set (13) without altering the variables within base data set (13) for future case creation and without copying the entire base data set (13) each time a case (20) needs to be created with changed variables.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

A CASE MANAGEMENT SYSTEM AND METHOD FOR MANAGING CASE MODELS OF PHYSICAL SYSTEMS

TECHNICAL FIELD OF THE INVENTION

The present invention relates to management software and, more particularly, to a software system and method for managing computer software models of physical systems.

BACKGROUND OF THE INVENTION

Highly complex physical systems require modeling to predict how the system will operate under a variety of conditions. Simulation modeling provides a tool to predict system operation, cost, and performance. Many highly complex physical systems use computer software programs to accomplish this modeling.

Systems for modeling the operation of a physical system differ, but typically the models require input of information from a base set of data. Typically this data defines the variables, assumptions, and other data that the physical system uses and produces. From the base set of data, a user can run a model to predict performance, cost, and other characteristics of the system in operation based on the variables in the base data. A user can then create different models of the system by altering variables in the base data and re-running the modeling software to create different case scenarios.

Each model of the system results in a case. In addition to running different models, users of these modeling systems often need to manipulate the different cases by, for example, storing the cases modeled, comparing the cases, and retrieving past cases. A case management system allows a user of such a modeling system to manage these individual cases and maintain a record of the various models and the associated results, such as performance and cost, of each case modeled.

Most physical systems have mechanisms built into them to allow an operator to change one or more of the variables in the system to try to improve performance or adjust to a change of other operating conditions. An effective case management system will allow a user of a modeling system to make changes to the source or base data in order to simulate a change in the operating conditions of the physical system.

This technique of changing variables in the base data presents problems in a typical database-type modeling system. One problem involves how to make changes to the base data for the case a user wants to model without affecting other cases made in the past, or cases made in the future. While a user could make a change to a variable directly to the base data, this change would affect all other users of the system because all cases run in the future must run with the changed variable. This change may also change the stored results of past cases modeled. Other users of the modeling system may not want to make that particular change to the base data. In fact, other users might decide to change a different variable altogether. Thus, making changes directly to the base data can lead to control problems.

When a user changes the base data, problems can arise with traceability of those changes. If a tracking system does not exist to alert a future user of the modeling system, a user might run a case without

realizing a change to the base data has occurred and make decisions based on a different set of variables than the user anticipated. Thus, making changes directly to the base data can also lead to traceability problems. In order to avoid this problem, many modeling systems prevent users from changing the base data.

One conventional approach to solving the problem of changing variables in the base data in a database-type structure of files requires a user to copy the entire base data base, and the associated files, and then make the change to the particular variable in the base data to create a case model with particular variable changes. This solution, however, leads to the problem of duplication of data. This replication of data requires additional user and computer time, and requires using additional computer storage space. Furthermore, while this solves the control over the base data problem, the problem of traceability remains. For example, if a user copies the base data file and stores this copied file separately from the original base data, no link exists between this copied data and the base data.

While a particular case model based on the copied data base with a changed variable may have positive results, that model does not link directly to the base data. This leaves other users no ready means of understanding what changes have been made to the base data. This type of copying of the base data prevents

readily sharing the information with other users of the model. The solution of copying the entire base data also leads to fragmentation of information because when the user makes a change to the base data, the case model from previous runs do not correlate with future case model runs.

Typical case management systems also do not provide for a hierarchy of base data assumptions where a user can have a particular set of assumptions that others can use and modify without affecting the original user's set of assumptions. For example, a user may want to use the set of variables another user has established, but make a change to one or more variables, while not changing the original set of variables borrowed from the other user. Typical management systems do not allow a first user to let other users of the modeling system access the first user's case models while, at the same time, protecting some or all of the first user's variables from being changed by those other users.

Another problem with current case management systems lies in the security associated with the cases. Current case management systems typically do not allow a user to define those variables in the user's base data that others cannot see (private assumptions), that others can see but not change (read-only assumptions), or that others can see and modify (write assumptions).

Current case management systems do not allow a user to easily replace portions of the base data (assumptions, variables or other data) with data from a particular case model. The created case might include many variable or assumption changes from the base data that the user would now want to replace in the base data, without, however, affecting the previously created cases. Current case management systems do not typically contain this feature.

SUMMARY OF THE INVENTION

The present invention provides a case management system and method that substantially eliminate or reduce disadvantages and problems associated with previously developed case management systems and methods.

More specifically, a case management system and method are provided that manage software case models that simulate operations of systems. Software cases are created by a user having access to a base data set that contains variables. The case manager software receives input from users to create cases that contain data from the base data set. The case manager software also creates cases directly from the base data set without the need to copy the base data for each created case. The case manager software also possesses the ability to receive input from users to create cases that allow the users to change variables in the base data and create cases. The created cases may contain a combination of unchanged variables from the base data, as well as changed variables from the base data set, without there being the need to alter the variables within the base data for future case creation. There is also no need to copy the entire base each time a need arises to create a case with changed variables.

The present invention includes an important technical advantage of allowing the modeling system user who accesses a base data set to change the data in the

base data set. Without copying a complete set of base data and without duplicating, the user may change the base data set to model "what-if" physical system scenarios without affecting the base data set for any other cases. The present invention allows a user to change a variable within the base data and run a case model, while the base data or base assumptions that all users have access to remains unchanged.

The present invention provides another technical advantage by allowing a user to run a case model and then to "promote" the variables of that case model into the base data. The present invention allows a user to run a case with variables different from the base data, and then modify the base data to agree with the data used in that particular case by promoting the case data to the base data. The present invention can incorporate a security system that defines which users can promote variables to base and which variables can or cannot be changed in the base data.

Yet another technical advantage of the present invention is the ability to limit the shareability of the base data. The present invention has a feature that can limit or expand access to the base data to individual users. This results in some users having a different view of base, but all users have access to the same case modeling and case management capabilities.

Still another technical advantage of the present invention lies in the ability to share cases. If a user makes a change to the base data and runs a case, other users can access that case and run other cases using that data.

A related technical advantage arises in the security associated with sharing cases. When a user creates a case with changes to the base data, the user can determine the level of security appropriate to the changes. By designating the variables/assumptions of the case as private, read-only public, writeable public, or some other appropriate designation, the user can designate the security level for the changes. Other users cannot see or modify private assumptions. Others sharing the case can see, but cannot change, the read-only public assumptions. Writable public assumptions can be seen and modified by other users.

The present invention provides another technical advantage with regard to traceability of changes to base. The present invention tracks case model changes so that the case variables that change and how those variables differ from the base variables can be determined.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying drawings in which like reference numerals indicate like features and wherein:

FIGURE 1 depicts lines of communication for use in one embodiment of the present invention;

FIGURE 2 illustrates a block diagram of one embodiment of the case manager of the present invention;

FIGURE 3 provides a block diagram of one embodiment of the create case operation of the present invention;

FIGURE 4 shows a block diagram of one embodiment of the select case operation of the present invention;

FIGURE 5 is a block diagram of one embodiment of the remove case operation of the present invention;

FIGURE 6 depicts a block diagram of one embodiment of the copy case operation of the present invention;

FIGURE 7 provides a block diagram of one embodiment of the move case operation of the present invention;

FIGURE 8 renders a block diagram of one embodiment of the compress case operation of the present invention;

FIGURE 9 shows a block diagram of one embodiment of the compare case operation of the present invention;

FIGURE 10 gives a block diagram of one embodiment of the promote case operation of the present invention; and

FIGURE 11 supplies a block diagram of one embodiment of the register change operation of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGUREs in which like numerals being used to refer to like and corresponding parts of the various drawings.

One application of the case management system of the present invention involves managing object oriented simulation models of physical systems. An example of a physical system modeled using object-oriented software may be a model for power system operations. In this type of model, simulating the power system includes building an object model of the power system. Accordingly, objects may model concrete things (e.g., people and computers), as well as abstract concepts, (e.g., numbers or geometrical concepts). The present invention can be utilized to manage these cases. The case management system of the present invention uses object oriented software objects in one implementation. The software objects of the present invention have been written in Small Talk, but could also be written in other software code languages including an object oriented code language such as C++.

FIGURE 1 shows object model diagram 12 of the interaction of case manager object 10 with user 11 and base data set 13. Base data set 13 includes assumptions and/or variables. Base data set 13 includes a case context 15 and a persistent collection 14.

In operation, user 11 interacts with case manager object 10 in some fashion that can include requesting certain functions or operations. The functions and operations case manager object 10 performs are discussed below in more detail. User 11 also has the option to modify base data set 13. Case manager object 10 either registers or retrieves case context 15 from persistent collection 14. Case context object 15 represents a specific case node created from base data set 13 that a user is manipulating. Persistent collection 14 represents a collection of software object cases created from base data set 13. Base data set 13 includes variables, assumptions, and other data available to all users 11. Base data set 13 can be controlled by a system administrator who defines the rules and access capabilities for any user 11. Subject to those rules, user 11 can make copies of base data set 13 for building particular models of the system.

FIGURE 2 shows case manager object 10 that includes application result object 22, case context 15 object, and case dictionary object 24 that further includes case node object 20, case change object 26 and audit log object 28. The case context object described above is retrieved from case dictionary 24. The application result object 22 represents the results from running a particular case model from a particular case context 15. Case dictionary object 24 stores all cases, or case node objects 20

created by user 11. Case change object 26 stores any changes made to case node 26. When user 11 makes a change to case node 26, case node 26 interacts with audit log object 28 to log various information, preferably including user 11 originating the change, the time the change occurred, the type of change, and the description of the change. Within the case management system, audit log object 28 creates an audit trail of changes made to case node objects 20.

The notation in FIGURE 2 represents FUSION notation as described in numerous texts, such as D. Coleman, Object-Oriented Development of Fusion Methods (1994). In all of the FIGURES, a box denotes an object, usually a software object, and a line connecting objects represents a functional relationship between those objects. A plus (+), asterisk (*), or a specific numeral (1, 2, 3, . . .) notation above the line represents the number of instances of that object. A plus indicates one or more instances of the connected object within that object, an asterisk represents zero or more instances of the connected object within that object, and a numeral represents that number of instances of the connected object within that object. For example, in FIGURE 2, case dictionary 24 can have one or more case node objects 20. Case node object 20 can either be created from base data set 13, or created from the data in an existing case node object 20. As the FUSION notation in FIGURE 3

shows, case node object 20 created from an existing case node object 20 can only have one case node object 20 as its source (source of data), but it can have many descendants that are created case node objects 20. In other words, case node object 20 can only have one parent, but can have as many children as users 11 decide to create. Each case node object 20 can have zero or more case changes. Furthermore, each case node 26 can log all of these changes, and thus, can log zero or more audit entries. Audit log object 28 thereby creates a chain of changes associated with each case node 26. This chain allows case manager object 10 to track the evolution of all case node objects 20.

The life cycle of case manager object 10 involves interacting with other objects, registering changes to case objects, and retrieving case objects. To interact with other objects, case manager object 10 must first create case node object 20. Case manager object 10 can then perform other interactions with objects including selecting, releasing, removing, copying, compressing, comparing, and promoting a case. Once the user creates a case, user 11 has the option to perform any of the other interactions as many times as user 11 chooses.

FIGURE 3 shows an object model diagram of the create case operation 30 that creates new case node object 20 from source data contained in either base data set 13 or an existing case node object 20. Create case operation

30 includes interaction between case manager object 10, case dictionary object 24, a new case node object 20 and audit entry object 28. To create a case, user 11 sends create case message 19 to create case node object 20 from case manager object 10. The data input from message 19, preferably includes the case name, case description and case source (the hierarchy parent).

Case manager object 10 assumes that the case name supplied is unique and occupies a unique name space. Case manager object 10 will modify case dictionary 24 to add new case node object 20 into source case dictionary 24 after new case node object 20 has been created. Case dictionary object 24 then creates new case node object 20 based on the input from operation 19 and creates an audit entry object to modify audit log object 28. Audit log object 28 modifies itself based on the information sent by case node object 20. Case manager object 10 then adds new case node object 20 to case dictionary 24 and returns a true if the operation of adding new case node object 20 is successful or a false if the operation is unsuccessful.

FIGURE 4 shows an object model diagram of select case operation 40 that selects existing case node object 20, in either read or write mode, from case dictionary 24. Select case operation 40 includes interaction between case manager object 10, case dictionary object 24, a consistent collection of case nodes 14 and case

context object 15. To select a case, user 11 sends select case message 19 to case manager object 10. The data input from message 19 includes the case name and the case access mode (read or write). Case manager object 10 assumes that the case name exists among existing case nodes object 20.

Case manager object 10 will select requested case node object 20 from case dictionary 24 if case node object 20 is available for the requested access mode of read or write. Case node object 20 determines its access status mode of either read or write. Case dictionary 24 then creates new case context 15 and populates the newly created case context 15 with selected case node object 20. Case manager object 10 then returns case context 15 to the user and returns a true if the operation of selecting new case node object 20 is successful or a false if the operation is unsuccessful.

FIGURE 5 shows an object model diagram of remove case operation 50 that removes an existing case node object 20 from case dictionary 24. The remove case operation deletes a case node from a case dictionary if case node object 20 is an end case (no dependents) and if the user possesses the appropriate authorizations to do this type of operation. Select case operation 40 includes interaction between case manager object 10, case dictionary object 24, a persistent collection of case nodes object 14 and audit object 15.

To remove a case, user 11 sends remove case message 19 to case manager object 10. Data input from message 19 includes the case name. Case manager object 10 assumes that the case name exists among existing case node objects 20 and that case node object 20 identified by that case name has no dependent case node objects 20. Case manager object 10 will remove requested case node object 20 from case dictionary 24 and creates an audit entry object to modify audit log object 28. Audit log object 28 modifies itself based on the information sent by case node object 20. Case manager object 10 then returns a true if the operation of removing case node object 20 is successful or a false if the operation is unsuccessful.

FIGURE 6 shows an object model diagram of the copy case operation 60 that copies existing case node object 20 to new case node object 20. The copy case operation copies the case node plus all data associated with that case node. This function allows a user to copy cases from other users to which the user may then make changes and run new cases. These changes to the copied case will not affect the original case in any way.

Copy case operation 40 includes interaction between case manager object 10, case dictionary object 24, a persistent collection of case nodes 14 and audit log object 28. To copy a case, user 11 sends copy case message 19 to case manager object 10. The data input

from message 19 includes the case name and the case destination. Case manager object 10 finds requested case node object 20 in case dictionary 24. Case manager object 10 will then create new case node object 20 and copy the information from requested case node object 20 into the new case node object 20. Case dictionary 24 then creates a new audit entry to record the case node copy event and the audit log object 28 modifies itself based on this information. Case manager object 10 then returns a true if the operation of copying case node object 20 is successful or a false if the operation is unsuccessful.

FIGURE 7 shows an object model diagram of the move case operation 70 that moves existing case node object 20 to new case node object 20. The move case operation replaces an existing destination case node with a target case node. Move case operation 70 includes interaction between case manager object 10, case dictionary object 24, a persistent collection of case nodes 14 and audit log object 28. To move a case, user 11 sends move case message 19 to case manager object 10. The data input from move case message 19 includes the case name and the case destination.

Case manager object 10 assumes that the target case name and the destination case name exist among existing case node objects 20. Case manager object 10 finds target case node object 20 from the persistent collection

of cases 14 within case dictionary 24. Case manager object 10 then removes target case node 20 from current source node 20 list of descendants, replaces the name of target case node 20 with the name of destination case node object 20, and add target case node 20 to the descendant list of the source node that includes that descendant name. Case dictionary 24 then creates a new audit entry to record the move case event, and the audit log object 28 modifies itself based on this information. Case manager object 10 then returns a true if the operation of moving case node object 20 is successful or a false if the operation is unsuccessful.

FIGURE 8 shows an object model diagram of the compress case operation 80 that compresses an existing set of dependent cases into the end case in the string of dependent case node objects 20. The compress case operation consolidates the changes made to case node object 20 at some point in the hierarchy to a different point upward in the hierarchy. The compress case allows a user to incorporate any selected number of changes to a case into a new case. Compress case operation 80 includes interaction between (a) case manager object 10, (b) case dictionary object 24, (c) a collection of case nodes 14, and (d) audit log object 28.

To compress a case, user 11 sends compress case message 19 to case manager object 10. The data input from compress case message 19 includes the begin case

name, the end case name, and the compress case name. Case manager object 10 assumes that the begin case name and the end case name exist among existing case node objects 20, and that the compress case name is unique among existing cases and does not currently exist. Case manager object 10 will find begin case node object 20 from the persistent collection of cases 14 within the case dictionary 24. Case manager object 10 then follows the inheritance path, defined by the nodes descending from begin case node object 20, to end case node object 20 specified in compress case message 19. Case manager object 10 then consolidates the changes made throughout the inheritance path and replace the state of end case node object 20 with the consolidated changes.

Case manager object 10 will then remove the case nodes in the inheritance path. Case dictionary 24 then creates a new audit entry to record the compress case event and audit log object 28 modifies itself based on this information. Case manager object 10 then returns a true if the operation of compressing case node object 20 is successful or a false if the operation is unsuccessful.

The inheritance path is defined by the original case created from the base data and cases created in a direct chain from that original case. In other words, an inheritance path contains an original case (created from base data or from base data with some variable

modifications) and a case created from that original case, and a case created from the secondary case and so on until a case exists from which no other cases have been created. The inheritance path follows singularly down the creation path so that there is only one case forming a link in the chain at every creation step. For example, if an original case created from base has two cases created from it, that means two inheritance paths now exist.

FIGURE 9 shows an object model diagram of the compare case operation 90 that allows a compare case object to access two or more case node objects 20 from within case dictionary 24 and compare the attributes of those two or more case node objects 20 with one another. Compare case operation 90 includes interaction between case manager object 10, case dictionary object 24, a persistent collection of case nodes 14, and compare case object 90 to create compare report 92. To compare two cases, user 11 sends compare case message 19 to compare case object 90. Compare case object 90 sends a compare case message to case manager object 10 informing the case manager of the source case name and the target case name.

Case manager object 10 assumes that the source case name and the target case name exist among existing case node objects 20. Case manager object 10 finds source case node 20 and the target case node from the persistent collection of cases 14 within case dictionary 24. Case

manager object 10 then provides the target case data and the source case data to compare case object 91 that then determines the differences between the two cases. Case manager object 10 creates compare report 92 into which the differences in the compared cases is recorded.

The compare case operation compares two existing cases and creates a report identifying the differences between the compared cases. While case manager object 10 itself does not perform this operation, the case manager's tracking of changes to case manager object 10 allows the comparison of the data, assumptions, variables, and results from two different cases. Thus, case manager object 10 makes it possible to access a separate software object (in an object-oriented scheme) to query the overall data system to compare two cases.

FIGURE 10 shows an object model diagram of the promote case operation 100 that promotes the variables in an existing case into base data set 13. Promote case operation 100 includes interaction between case manager object 10, the case dictionary object 24, a persistent collection of case nodes 14 and an audit log object 28. To promote a case, user 11 sends promote case message 19 to case manager object 10. The data input from promote case message 19 includes the case name of the case to be promoted. Case manager object 10 assumes that the promote case name exists among existing case node objects 20.

Case manager object 10 will find promote case node object 20 from the persistent collection of cases 14 within case dictionary 24. Case manager object 10 then promotes the changes associated with case node object 20 into the base data set 13. This process involves identifying all changes that have occurred in case node object 20 and changing any base variables/assumptions/data that have been changed. Case dictionary 24 then creates a new audit entry to record the promote case event and the audit log object 28 modifies itself based on this information. Case manager object 10 then returns a true if the operation of promoting case node object 20 is successful or a false if the operation is unsuccessful.

Promote case operation 100 allows a user to impose the changes in the data made to create a particular case (including any changes along the cases inheritance path) onto the base data. Thus, the base data for every user from this point forward will have the promoted data. This does not affect previously built cases, but works prospectively to affect all future cases. A security system can establish which users can promote changes and what variables these users can actually change.

FIGURE 11 shows an object model diagram of register change operation 110 that registers the variable changes made to existing case node object 20. Case manager object 10 makes a change to case node object 20, each case node

object 20 performs register change operation 110 and informs audit object log 28. Audit log object 28 then logs the change to case node object 20. Case manager object 10 then returns a true if the operation of registering case node object 20 is successful or a false if the operation is unsuccessful.

Other specific operations could be included to further enhance the case management system including a retrieve change operation where the case manger retrieves a set of changes made to a case node or a set of changes made to the base data.

In operation, the case management system and method of the present invention supports numerous cases where a case can represent a particular computer model of a physical system. The modeling of the physical system begins based on a base data set including variables, assumptions, and other data. All users have access to the base data. However, the system administrator can limit the amount of access of certain users to case management system objects and the associated base data. For example, the systems administrator can give user A unlimited access to the entire base when modeling cases, but can restrict user B from access to one or more specific variables or assumptions in the base. However, each user case, regardless of the amount of access the user has to the base data, inherits the same capabilities as every other case.

The present invention allows a user to specify a set of changes to the base model state that represents the system at some reference point in time to create a case representing a different model of the system. A user can make any number of changes to the base data that the user can access. The case management system allows a user to perform "what-if" analysis by creating a study case from the base model state (root case) or create a study case from an existing case (branch or leaf case). The case management system records and tracks the changes made to the base data in subsequent cases in a case dictionary that represents a repository for all changes associated with the case inheritance chain. The net state of a case is determined by applying the ordered set of changes from the root case to the case along the case inheritance path.

The case manager also provides controlled multi-user access to the individual cases. Only one user can create a case (each case may be write protected), but an unlimited number of users can read a case. Each user can then modify the model state represented by the case independent of any other users without affecting the case from which the modified case was built. If a user modifies a case that already has dependent cases built from it, the case manager object notifies the dependent cases that changes have been made.

Modifications to a case may be made even when a dependent case is currently in use. The case manager will notify the users of the cases affected that the results of simulations based on those modified case nodes are now invalid. Thus, the case nodes have inherited the changes and the previous results from those cases in their previous form are now invalid. The user can access audit log object 28 to determine what changes to the case nodes have occurred.

Each case created, either from the base data or from the data in existing cases, gets added to the case dictionary that stores the created cases. Each case that gets created from a parent (existing) case become a part of the inheritance path of that case. Thus, an inheritance path includes the original case created from base data, the case created from that original case, the case created from the second case, and so on until no other cases have been created in that chain. The case manager also allows a user to consolidate all changes into a destination case by compressing the case tree structure into a single case to capture all the changes to base that have occurred on a particular inheritance path.

The case manager further allows a user to compare two cases or a case to its base model state and report the differences. A user can also select all or some of

the data from a particular case to be promoted to the base data.

The case manager also maintains an audit trail of all modifications made to the case. The audit object logs interactions with a case to include what was changed, when it was changed and who changed it. The case manager allows a user to view the change events made (the audit trail) of any case. The audit trail that belongs to the root or original case will be removed from the system upon deletion of the case root and all its dependents.

The case manager allows a user to store and retrieve a case. The case manager also stores output result objects containing the study outcomes of application models. In other words, the case manager will store the simulation model results based on case information for every case model that gets run as a simulation.

A user may also delete a study case if the study case has no dependent cases. The present invention prevents deletion of a case with dependents to avoid deleting the dependent cases. By preventing cascade deletes, the case manager ensures that dependent cases built off of existing cases are not lost.

The case management system can be written in SmallTalk object oriented code. The management of cases allows the user to create cases, each case representing one model of a system, from a set of base

data/assumptions/variables. Each case can be modified by the user. The present invention allows other users to see cases (results for a particular modeling of the system). Case management also allows other users to change public (non-private) variables/data/assumptions to get a different model of the system and store that case in the case management system. A user can also change the base assumptions by "promoting" the changes from a particular case to the base data.

In summary, the present invention provides a case management system and method for managing case models of physical systems that allows a user to create cases with changed base variables without copying the entire base and without altering the base data for future. The case manager software receives input from users to create cases containing data from the base data. The case manager software also creates cases directly from the base data without copying the base data for each case created. The case manager software can also receive input from users to create cases that allows the users to change variables in the base data and creation of cases containing a combination of unchanged variables from the base data and changed variables from the base data without altering the variables within the base data for future case creation and without copying the entire base each time a case needs to be created with changed variables.

Although the present invention has been described in detail, it should be understood that various changes, substitutions and alterations can be made hereto without departing from the spirit and scope of the invention as described by the appended claims.

WHAT IS CLAIMED IS:

1. A case management system, for implementation on a computer, for managing cases representing models of an operation system where the cases are created by a user having access to a base data set comprising a plurality of variables, said case management system comprising:

a case manager module comprising instructions for receiving input from a plurality of users for creating a plurality of cases comprising data from said base data set, said case manager module further comprising instructions for creating said plurality of cases directly from said base data set without copying said base data set for each case created;

said case manager module further comprising instructions for receiving input for creating a plurality of cases, said plurality of cases comprising a plurality of unchanged variables from said base data set and a plurality of changed variables from said base data set, said receiving input occuring without altering said plurality of variables within said base data set for creating future cases.

2. The system of Claim 1 wherein said case manager module further comprises:

instructions for creating a plurality of cases based on previously created cases, said case manager module further comprising instructions for creating said plurality of cases from data contained within previously created ones of said plurality of cases, and

further wherein said case manager comprises instructions for creating a plurality of cases from both unchanged data from said previously created ones of said plurality of cases and changed data from said previously created ones of said plurality of cases without altering values of variables within said previously created cases.

3. The system of Claim 2, wherein said case manager module further comprises:

instructions for building a plurality of inheritance paths for ones of said plurality of cases created from previously created ones of said plurality of cases such that each of said plurality of inheritance paths comprising an original case, said original case created from said base data set, and a plurality of direct cases, said direct cases created in a direct chain from said original case.

4. The system of Claim 3, wherein said case manager module further comprises instructions for running a model simulation of a physical system operation, said model simulation based on the data contained within a plurality of created cases.

5. The system of Claim 4, wherein said case manager module further comprises instructions for allowing a plurality of users to read and modify data within at least one of said plurality of cases.

6. The system of Claim 4, wherein said case manager module further comprises a case manager object, said case manager object written in object-oriented software code and the software system further comprises a software system written in object-oriented code.

7. The system of Claim 6, further comprising:

a case dictionary object for receiving input from and sending output to a plurality of software modules;

a plurality of case objects comprising said plurality of created cases;

a case context object comprising the specific case node object the user uses to run a simulation;

a persistent case collection object containing all case changes; and

an audit log object for logging the various operations performed on said plurality of cases, for tracking changes to said plurality of cases, and for tracking changes to said base data set.

8. The system of Claim 7, wherein said case manager object further comprises case object creation instructions for creating a case object, said case object creation instructions comprising instructions for:

receiving a create case message input from a user with data including a case name, a case description, and a case source;

creating a case object based on input received;

adding said created case object to said case dictionary object;

sending the information concerning the creation of the case to said audit log object, said audit log

object comprising instructions for logging the case creation event; and

returning a true to the user if the creation event was successful and returning a false to the user if the creation event is unsuccessful.

9. The system of Claim 7, wherein said case manager object further comprises case objection selection instructions for selecting a case object, said case object selection instructions comprising instructions for:

receiving a select case message input from a user with data including the existing case name and existing case access mode;

sending the input information to said case dictionary object, said case dictionary object comprising instructions for selecting said existing case object if said case object is available for the requested access mode, said case dictionary further comprising instructions for creating a new case context object and populating said new case context object with the selected case object data;

said case dictionary object further comprising instructions for returning said selected existing case object to an original location in said case dictionary object; and

returning a true if said selection event is successful and returning a false if said selection event is unsuccessful.

10. The system of Claim 7, wherein said case manager object further comprises case object removal instructions for removing a case object, said case object removal instructions comprising instructions for:

receiving a remove case message input from a user with data including a case name for a case;

sending the input information to said case dictionary object, said case dictionary object comprising instructions for finding the case object;

removing said case object from said case dictionary object; and

returning a true if removing said case object is successful and returning a false if removing said case object is unsuccessful.

11. The system of Claim 7, wherein said case manager object further comprises case object moving instructions for moving a case object, said case object moving instructions comprising instructions for:

receiving a move case message input from a user with data including a name and a destination for a case object to be moved;

releasing any control activity associated with said case object to be moved;

removing said case object to be moved from a current case inheritance path associated with said case objects to be moved and placing said case object to be moved in a destination case object inheritance path; and

returning a true if moving said case object to be moved is successful and returning a false if moving said case object to be moved is unsuccessful.

12. The system of Claim 7, wherein said case manager object further comprises case object copying instructions for copying a case object, said case object copying instructions comprising instructions for:

receiving a copy case message input from a user with data including a name and a destination for an existing case object;

finding said existing case object in a case dictionary object;

copying said existing case object into a destination case object and placing said destination case object in said case dictionary object; and

returning a true if copying said existing case object is successful and returning a false if copying said existing case object is unsuccessful.

13. The system of Claim 7, wherein said case manager object further comprises case object compressing instructions for compressing a case object, said case object compressing instructions comprising instructions for:

receiving a compress case message input from a user with data including a beginning case name for a case object to be compressed, an ending case name for said case object to be compressed, and compressed case name for said case object to be compressed;

finding a begin case object for said case object to be compressed in said case dictionary object;

finding an end case object for said case object to be compressed in said case dictionary object;

finding all case objects in an inheritance path between said begin case object and said end case object;

combining a state of said begin case object with the states of said inheritance path case objects and said end case object;

replacing said state of the end case object with said combined state;

removing said begin case object and all case objects in said inheritance path except the case object containing the combined state; and

returning a true if the compress event is successful and returning a false if the replace compression event is unsuccessful.

14. The system of Claim 7, wherein said case manager object further comprises case object comparing instructions for comparing a source case object with a target case object, said case object comparing instructions comprising instructions for:

receiving a compare case message input from a user with data including the source case name and target case name;

finding the source case object and the target case object in the case dictionary object;

comparing the source case object data to the target case object data;

creating a report documenting the differences in data state of the source case object and the target case object; and

returning a true if the comparison event is successful and returning a false if the comparison event is unsuccessful.

15. The system of Claim 7, wherein said case manager object further comprises case object registering instructions for registering a change to a case object, said case object registering instructions comprising;

performing a change to a case object in the case dictionary and each case object that is changed sends a message to the auditor object that informs the

auditor object of the changes made to the case object, the auditor object then logs the change; and

returning a true if the register change event is successful and returning a false if the register change event is unsuccessful.

16. The system of Claim 7, wherein said case manager object further comprises case object promoting instructions for promoting a case object, comprising instructions for:

receiving a promote case message input from a user with data including a name of a case to be promoted;

finding said case object to be promoted in said case dictionary;

comparing the differences between the data contained in said case object to be promoted with the data contained in base; and

incorporating any differences in the data from said case object to be promoted into the base in a manner that modifies base for any cases created after promoting said case object to be promoted; and

returning a true if promoting said case object to be promoted is successful and returning a false if promoting said case object to be promoted is unsuccessful;

thereby allowing the promotion of changes to variables in the base data to the values contained within

a particular case object without modifying case objects created prior to the time of promoting said case object to be promoted.

17. The system of Claim 16, wherein said case manager object further comprises instructions for adding variables to said base data set if the variables from said case object to be promoted did not previously exist in said base data set, and to change variables in the base if the variables to be promoted from said case object to be promoted existed in base prior to promoting said case object to be promoted.

18. The system of Claim 9, wherein the case manager object further comprises interfacing with a security software module to designate which users may promote case objects to said base data set and to designate which variables may be changed within said base data set.

19. A method for managing a plurality of model cases, said model cases derived from a base data set, where said model cases simulate operation of a plurality of physical systems, the method comprising the steps of:

creating a plurality of cases containing data from said base data set without copying the base data set for each created case;

creating a plurality of cases containing unchanged variables from said base data set and changed variables from the base data without altering the variables within the base data for future case creation;

creating a plurality of cases based on previously created cases;

creating a plurality of cases from data contained within previously created cases containing unchanged data from previously created cases and changed data from previously created cases without altering the variable values within the previously created cases;

building an inheritance path for cases created from previously created cases such that each inheritance path contains an original case created from the base data and cases created in a direct chain from that original case;

storing the created cases in a case dictionary software module;

creating a case context software module comprising the data from a specific created case, the case context

used by a user to run a simulation of the system operation; and

logging operations performed on the created cases in an auditor software module.

20. The method of Claim 19, further comprising the steps of

creating the case dictionary software module from object orient code to create a case dictionary object;

creating the case context software module from object oriented code to create a case context object;

creating the auditor software module in object oriented code to create an auditor object;

and creating the plurality of cases with object oriented code to create a plurality of case objects.

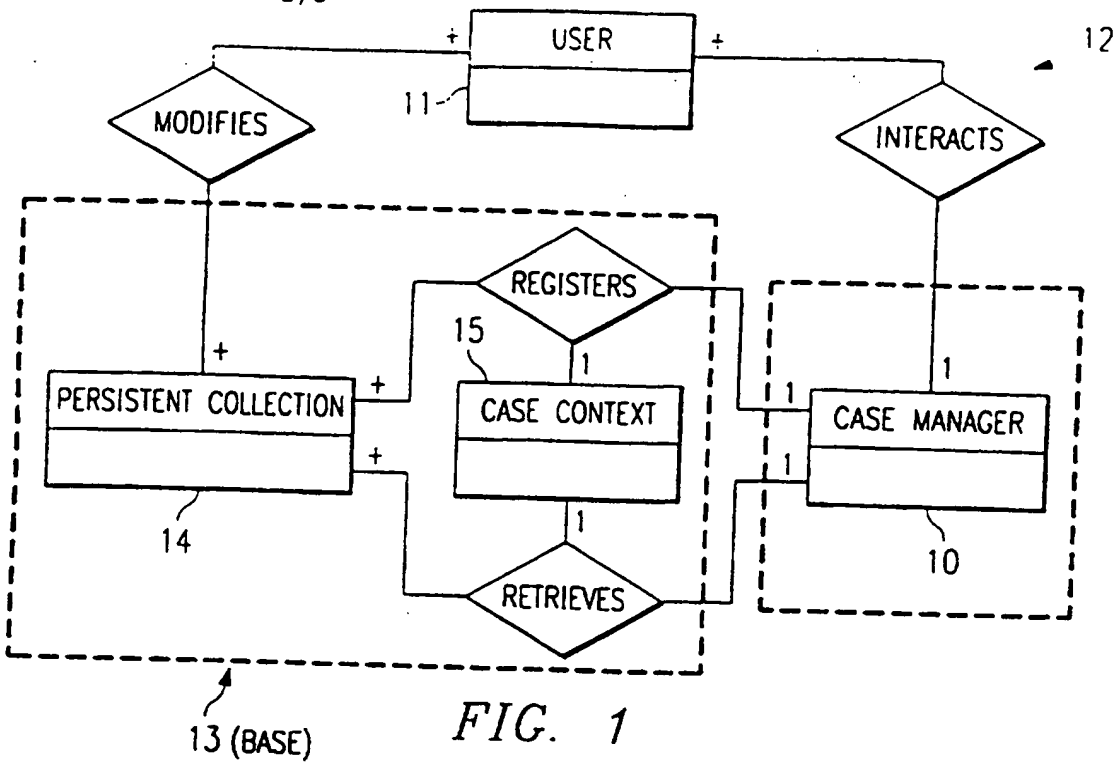

USER
11
12
MODIFIES
INTERACTS
REGISTERS
15
PERSISTENT COLLECTION
CASE CONTEXT
CASE MANAGER
14
10
RETRIEVES
13 (BASE)

FIG. 1

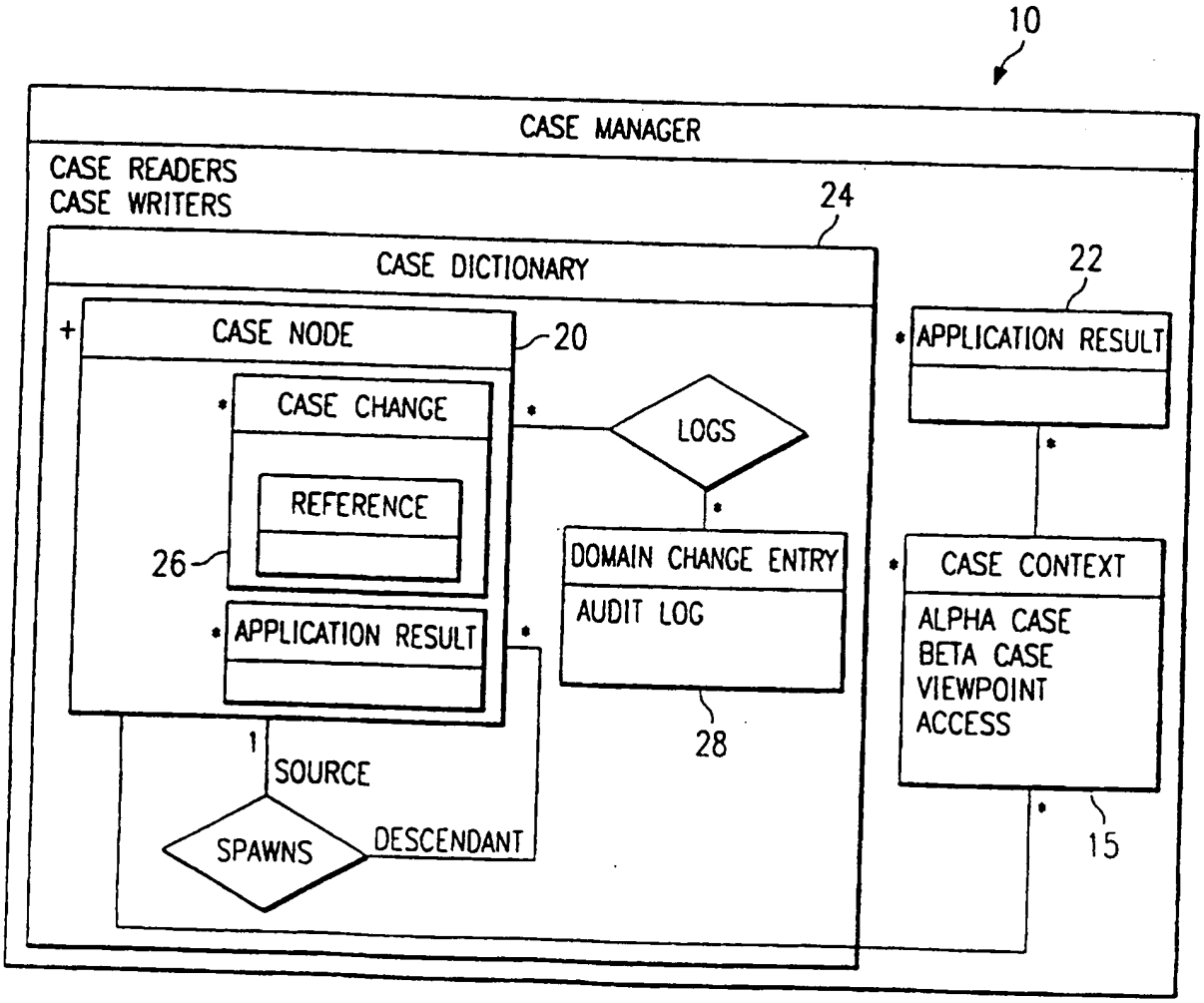

10
CASE MANAGER
CASE READERS
CASE WRITERS
24
CASE DICTIONARY
22
CASE NODE
20
APPLICATION RESULT
CASE CHANGE
LOGS
REFERENCE
26
DOMAIN CHANGE ENTRY
AUDIT LOG
28
APPLICATION RESULT
CASE CONTEXT
ALPHA CASE
BETA CASE
VIEWPOINT
ACCESS
SOURCE
SPAWNS
DESCENDANT
15

FIG. 2

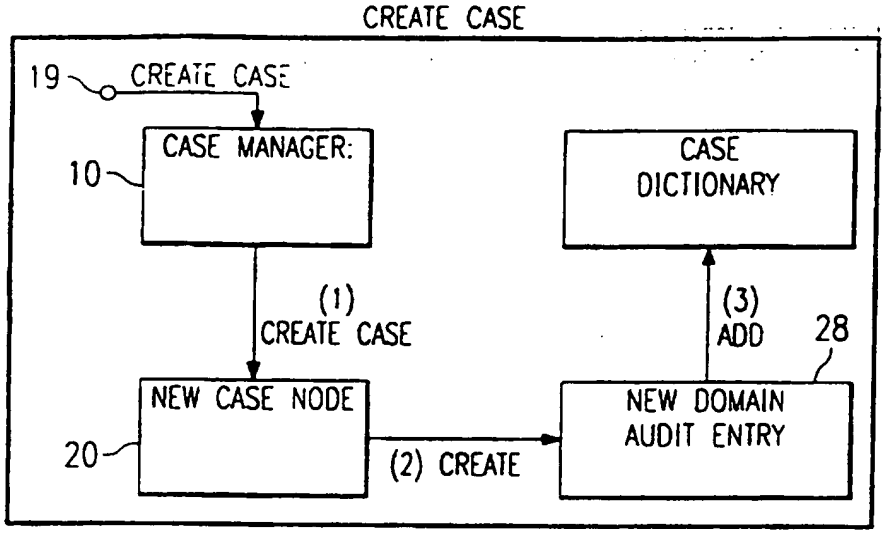


FIG. 3

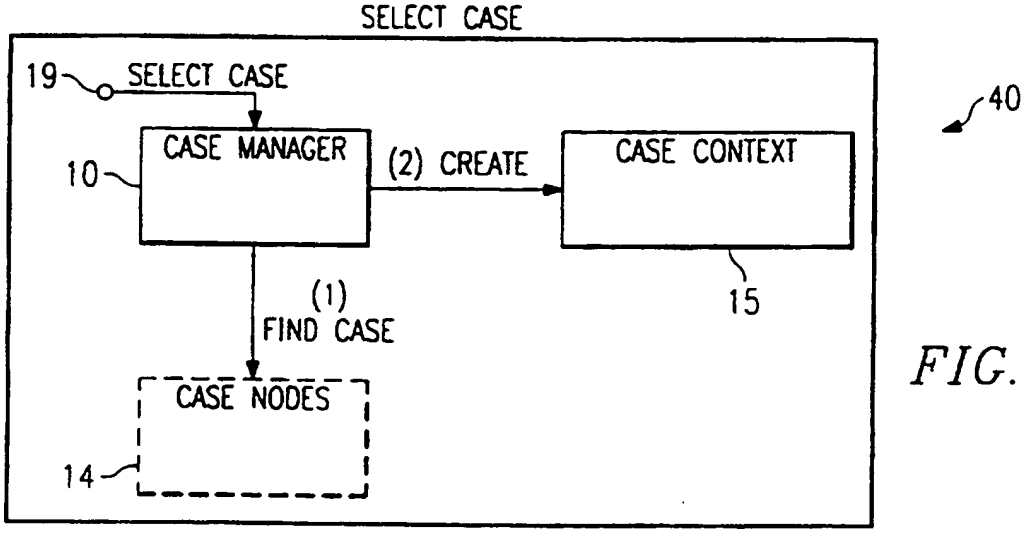


FIG. 4

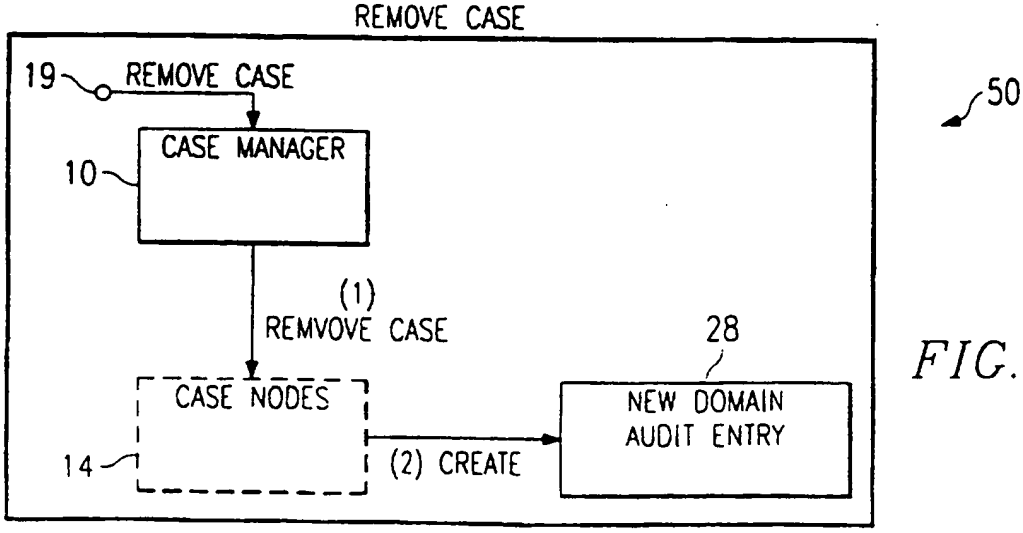


FIG. 5

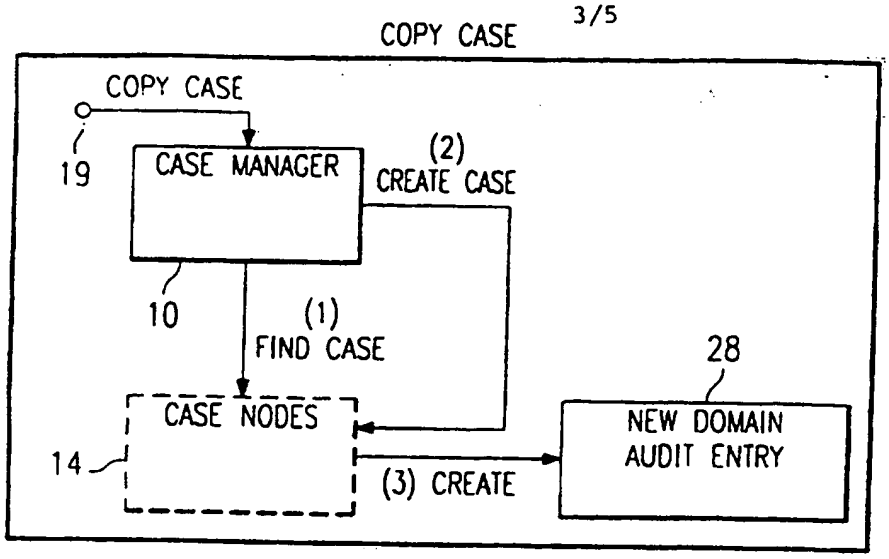


FIG. 6

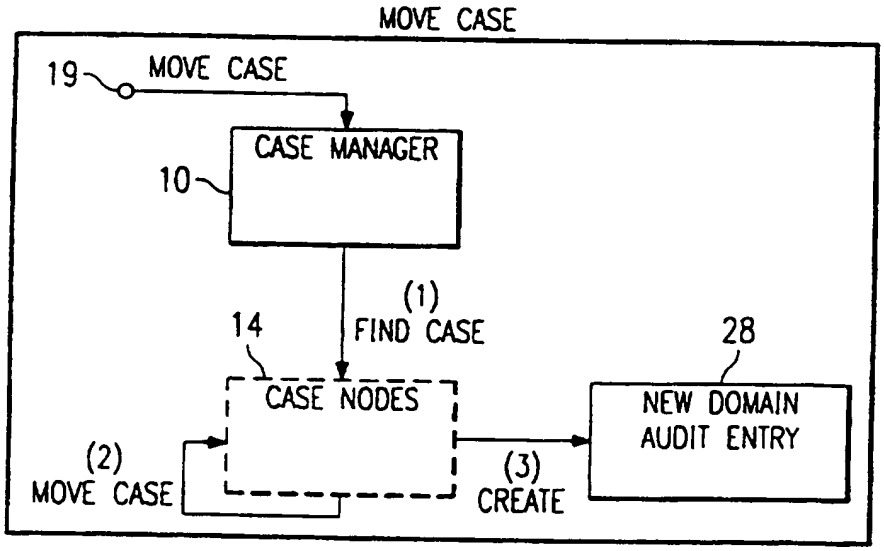


FIG. 7

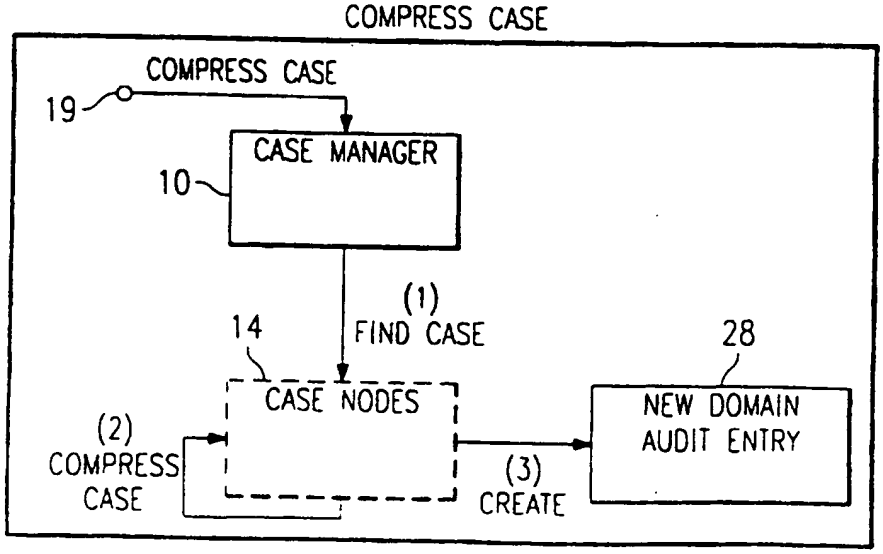


FIG. 8

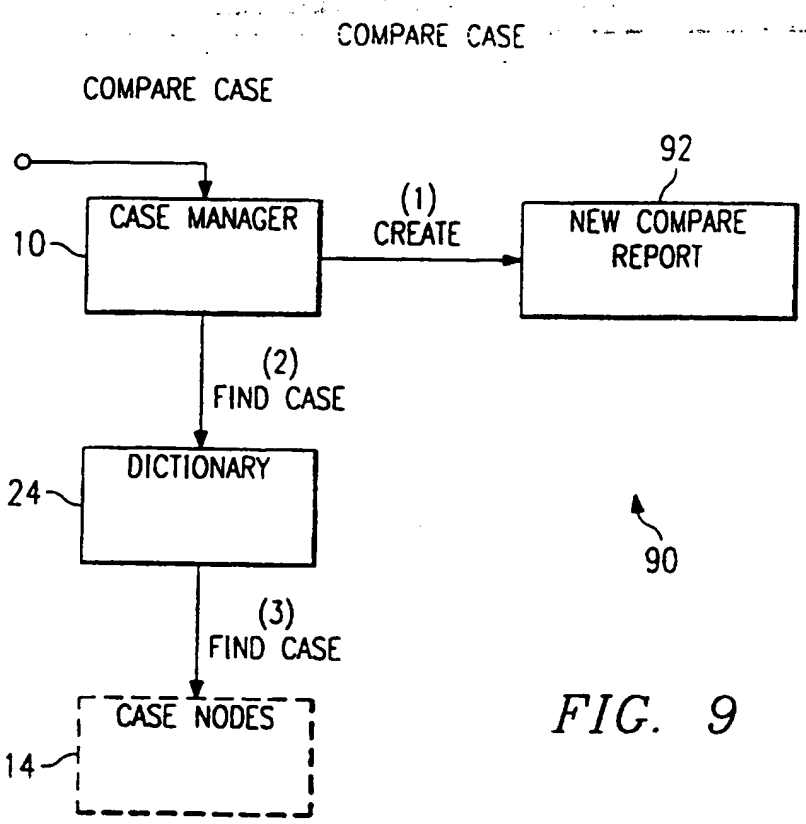

COMPARE CASE
COMPARE CASE
92
CASE MANAGER
10
(1)
CREATE
NEW COMPARE
REPORT
(2)
FIND CASE
DICTIONARY
24
90
(3)
FIND CASE
CASE NODES
14

FIG. 9

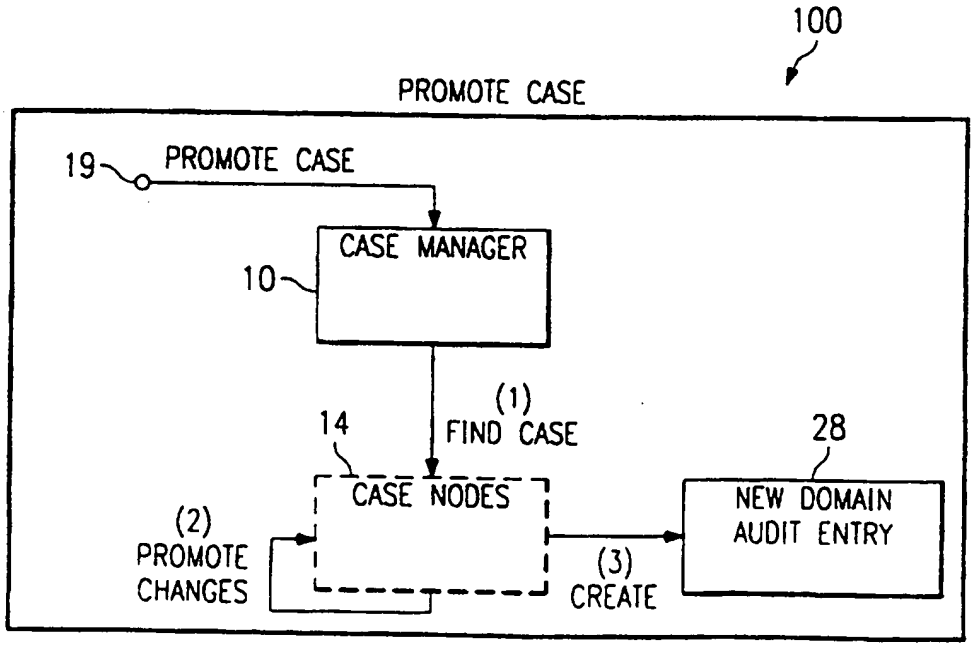

100
PROMOTE CASE
19
PROMOTE CASE
CASE MANAGER
10
(1)
FIND CASE
14
28
CASE NODES
NEW DOMAIN
AUDIT ENTRY
(2)
PROMOTE
CHANGES
(3)
CREATE

FIG. 10

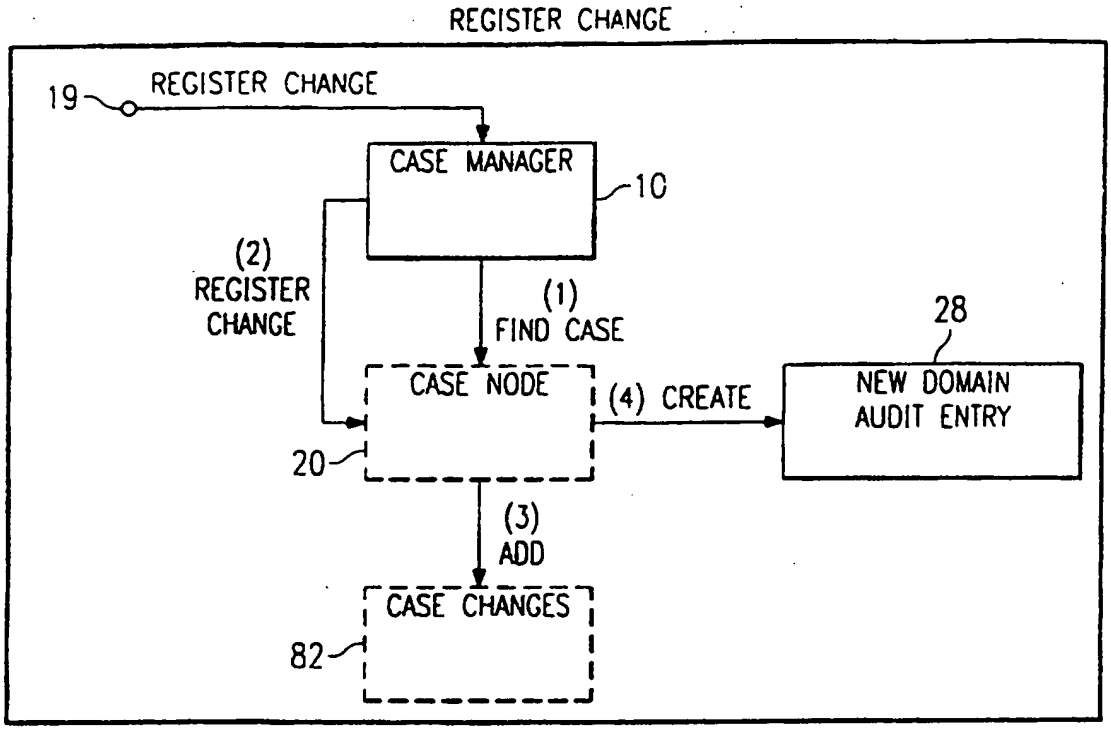

REGISTER CHANGE
19
REGISTER CHANGE
CASE MANAGER
10
(2)
REGISTER
CHANGE
(1)
FIND CASE
28
CASE NODE
(4) CREATE
NEW DOMAIN
AUDIT ENTRY
20
(3)
ADD
CASE CHANGES
82
110

FIG. 11

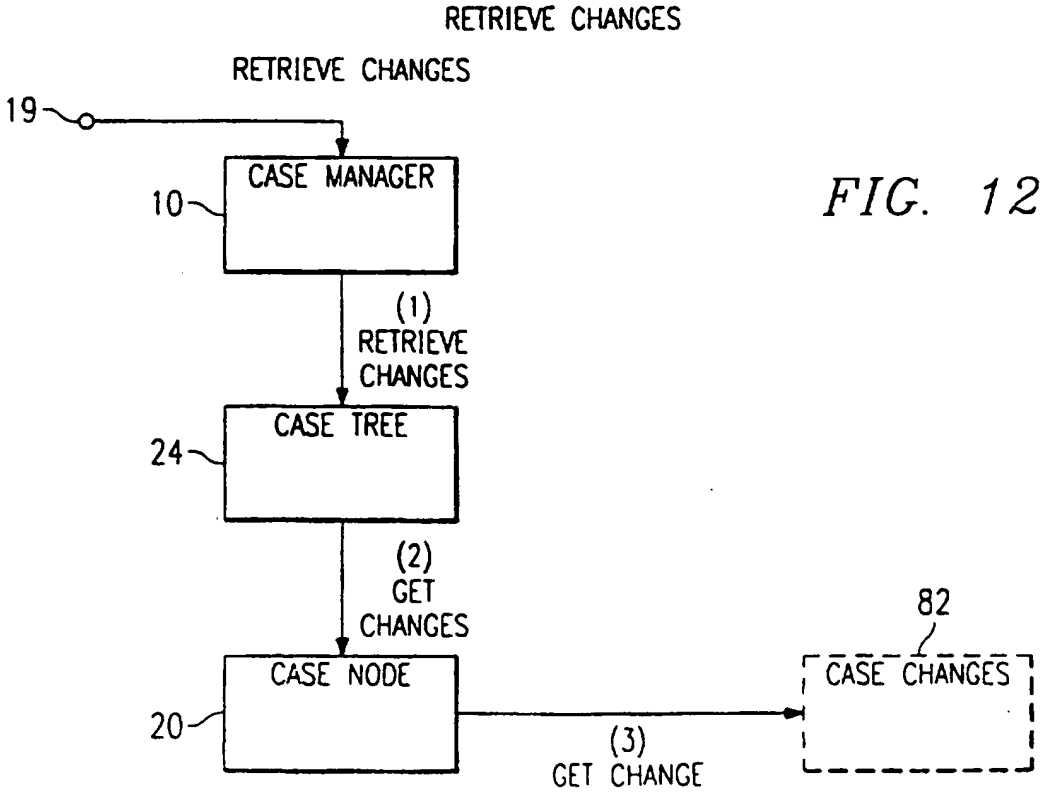

RETRIEVE CHANGES
RETRIEVE CHANGES
19
CASE MANAGER
10
(1)
RETRIEVE
CHANGES
CASE TREE
24
(2)
GET
CHANGES
82
CASE NODE
CASE CHANGES
20
(3)
GET CHANGE

FIG. 12

INTERNATIONAL SEARCH REPORT

[redacted] International Application No
PCT/US 97/03473

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JOURNAL OF SYSTEMS ENGINEERING, vol. 3, no. 4, 1993, UK, pages 184-190, XP002035190 EGE R K: "database support for object oriented simulation" see page 185, column 1, line 10 - line 45 ---	1,19
A	PROCEEDINGS OF THE WINTER SIMULATION CONFERENCE, LOS ANGELES, DEC. 12 - 15, 1993, no. -, 12 December 1993, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 552-559, XP000479554 ROVIRA M ET AL: "THE CONCEPT OF VIEWS IN SIMULATION" see page 552, column 2, line 14 - line 40 --- -/--	1,19

[X] Further documents are listed in the continuation of box C. [ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 July 1997	22.07.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Guingale, A

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IEEE COMMUNICATIONS MAGAZINE, vol. 32, no. 3, 1 March 1994, pages 64-69, XP000442188 SAULNIER E T ET AL: "SIMULATION MODEL REUSABILITY" see page 68, column 2 - page 69, column 1, line 46; figure 4 ---	1,19
A	ACM TRANSACTIONS ON MODELING AND COMPUTER SIMULATION, vol. 1, no. 3, 1 July 1991, pages 195-218, XP000290594 ZEIGLER B P ET AL: "MODEL BASE MANAGEMENT FOR MULTIFACETTED SYSTEMS" see page 198, line 6 - line 18 see page 199, line 41 - line 45; figure 3 -----	1,19